МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева (Самарский университет)»

Институт информатики, математики и электроники

Факультет информатики   
Кафедра программных систем

**ОТЧЕТ**

к лабораторной работе по дисциплине

«Нейронные сети глубокого обучения»

Студент Е. Г. Плешаков

Студент О.В. Ширяева

Преподаватель А. Н. Жданова

Самара 2021

Содержание

[1 Постановка задачи 3](#_Toc54911415)

[2 Исходный код 4](#_Toc54911416)

[2.1 Последовательное выполнение программы 4](#_Toc54911417)

[2.1 Параллельное выполнение программы 4](#_Toc54911418)

[3 Результат работы 6](#_Toc54911419)

[4 Вывод 8](#_Toc54911420)

Постановка задачи

Цель лабораторной работы: реализовать нейронную сеть для классификации изображений.

Классификация (рубрикация) изображений представляет собой отнесение изображения к одной из нескольких категорий (рубрик) на основании его содержания. В одной из своих многочисленных постановок задача распознавания образов трактуется как отнесение предъявленного объекта по его формальному описанию к одному из заданных классов. Важно, что классы (категории, рубрики) в задаче классификации являются пересекающимися подмножествами, это значит, что текст может быть отнесен к нескольким категориям одновременно.

* имеется множество категорий (классов, рубрик): ;
* имеется множество документов: ;
* существует неизвестная целевая функция: ;
* имеется некоторая начальная коллекция размеченных изображений  , для которых известны значения . Её делят на «обучающую» и «тестировочную» части. Первая используется для обучения классификатора, вторая – для проверки качества его работы.
* необходимо создать классификатор , максимально близкий к , который бы выдавал степень подобия.

1. Описание архитектуры сети, метода обучения
2. Вычислительные эксперименты
3. Результат работы
4. Вывод

В результате вычислительных экспериментов было обнаружено что оптимальной структурой для сети, классифицирующей изображения размером 128х128 точек, будет 4 слоя, включая входной и выходной. При удалении одного слоя резко уменьшается точность распознавания. При добавлении дополнительного слоя точность незначительно увеличивается, но намного больше вырастает время обучения.

Приложение А  
Код программы

**Файл create\_cnn.py**

# Importing the Keras libraries and packages

from keras.layers import Conv2D

from keras.layers import Dense

from keras.layers import Flatten

from keras.layers import MaxPooling2D

from keras.models import Sequential

from tensorflow import keras

# Initialising the CNN

model = Sequential()

# Step 1 - Convolution

model.add(Conv2D(32, (3, 3), input\_shape=(128, 128, 3), activation='relu'))

# Step 2 - Pooling

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Adding a second convolutional layer

# model.add(Conv2D(32, (3, 3), activation='relu'))

# model.add(MaxPooling2D(pool\_size=(2, 2)))

# Step 3 - Flattening

model.add(Flatten())

# Step 4 - Full connection

# model.add(Dense(units=256, activation='relu'))

model.add(Dense(units=64, activation='relu'))

model.add(Dense(units=1, activation='sigmoid'))

# Compiling the CNN

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Part 2 - Fitting the CNN to the images

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale=1. / 255,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True)

test\_datagen = ImageDataGenerator(rescale=1. / 255)

batch\_size = 32

training\_set = train\_datagen.flow\_from\_directory('dataset/training\_set',

target\_size=(128, 128),

batch\_size=batch\_size,

class\_mode='binary')

test\_set = test\_datagen.flow\_from\_directory('dataset/test\_set',

target\_size=(128, 128),

batch\_size=batch\_size,

class\_mode='binary')

import os

cp\_callback = keras.callbacks.ModelCheckpoint(filepath=os.curdir,

save\_weights\_only=True,

verbose=1)

model.fit(training\_set,

steps\_per\_epoch=8000 // batch\_size,

epochs=25,

validation\_data=test\_set,

validation\_steps=2000 // batch\_size,

callbacks=[cp\_callback])

model.save("keras\_cnn")

print("Model saved")

**Файл create\_cnn.py**

import os

import matplotlib.pyplot as plt

import matplotlib

import numpy as np

import pandas as pd

import tensorflow as tf

from keras.preprocessing import image

from keras.preprocessing.image import ImageDataGenerator, load\_img

model = tf.keras.models.load\_model("keras\_cnn")

print("Model loaded")

train\_datagen = ImageDataGenerator(rescale=1. / 255,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True)

test\_datagen = ImageDataGenerator(rescale=1. / 255)

training\_set = train\_datagen.flow\_from\_directory('dataset/training\_set',

target\_size=(128, 128),

batch\_size=32,

class\_mode='binary')

test\_set = test\_datagen.flow\_from\_directory('dataset/test\_set',

target\_size=(128, 128),

batch\_size=32,

class\_mode='binary')

test\_filenames = os.listdir("dataset/control\_set")

test\_df = pd.DataFrame({

'filename': test\_filenames

})

nb\_samples = test\_df.shape[0]

sample\_test = test\_df.head(18)

sample\_test.head()

plt.figure(figsize=(6, 6))

for index, row in sample\_test.iterrows():

filename = row['filename']

img = load\_img("dataset/control\_set/" + filename, target\_size=(128, 128))

test\_image = image.load\_img("dataset/control\_set/" + filename, target\_size=(128, 128))

test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis=0)

result = model.predict(test\_image)

if result[0][0] == 1:

prediction = 'dog'

else:

prediction = 'cat'

category = prediction

axes = plt.subplot(6, 4, index + 1)

plt.imshow(img)

plt.xlabel(format(category))

axes.set\_yticks([])

axes.set\_xticks([])

plt.tight\_layout()

plt.show()